Business Intelligence and Column-Oriented Databases

Kornelije Rabuzin
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
kornelije.rabuzin@foi.hr

Nikola Modrušan
NTH Mobile,
Međimurska 28, 42000 Varaždin, Croatia
nmodrusa@hotmail.com

Abstract. *In recent years, NoSQL databases are becoming more and more popular. We distinguish several different types of such databases and column-oriented databases are very important in this context, for sure. The purpose of this paper is to see how column-oriented databases can be used for data warehousing purposes and what the benefits of such an approach are. HBase as a data management system is used to store the data warehouse in a column-oriented format. Furthermore, we discuss how star schema can be modelled in HBase. Moreover, we test the performances that such a solution can provide and we compare them to relational database management system Microsoft SQL Server.*

Keywords. Business Intelligence, Data Warehouse, Column-Oriented Database, Big Data, NoSQL

1 Introduction

In the past few years NoSQL databases are becoming more and more popular. Some features that are crucial for the relational data model do not seem to be appropriate for today's applications. Applications that are built today require much better response time, they do not need a strict ACID (Atomicity, Consistency, Isolation, Durability) [18], versioning is actually a desirable feature, the volumes of data are becoming very large etc. Since the relational data model does not cope with these issues very well, NoSQL databases are here to fill in the gap.

When we talk about NoSQL databases, we have to have in mind that four different types can be distinguished; key value, document-oriented, column-oriented and graph databases.

Document-oriented databases store data in documents, but documents are not in a form that we are used to. Document does not mean a .doc (or similar) file, but usually JSON is used as a data format because it can be easily read and interpreted [15]. In JSON, objects are crucial. An object is in fact a nested string/value structure (Fig. 1). A value can be another object, a list of values etc. One of the most popular document-oriented database systems is MongoDB.

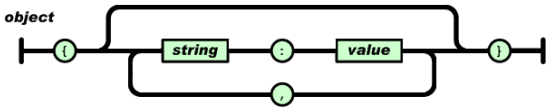


Figure 1. JSON object [15]

Graph databases, on the other hand, rely on some segment of the graph theory. They are good to represent nodes (entities) and relationships among them. This is especially suitable to analyze social networks and some other scenarios.

Key value databases are important as well for a certain key you store (assign) a certain value. Document-oriented databases can be treated as key value as long as you know the document id. Here we skip the details as it would take too much time to discuss different systems [21].

Column-oriented databases such as HBase are exactly that, column oriented. This means that column values are stored together (instead of rows). Since queries that are used to extract data usually retrieve only some columns from a table (queries such as SELECT * FROM table should be avoided), column-oriented solutions should be much faster. This paper explores how column-oriented databases (HBase in our case) can be used for data warehousing purposes and what are the benefits of such an approach.

The paper is organized as follows. First section explores HBase as a column-oriented database representative. The second section presents several major findings in the field of data warehousing for NoSQL databases. The third section discusses an example scenario that is then implemented as well as some experimental results. Finally, the conclusion is presented.

2 HBase storage system

Data warehouses have been used for a long time. Their main purpose is to integrate data from heterogeneous sources in order to build a unified and

cleaned structure called a data warehouse. Many good sources can be found on the subject of facts and dimensions (for example, [16], [13], [4], [19] or [2]) and we will not go into detail here.

Collection of data can be stored by means of different storage models. Standard data warehouses are mostly based on the relational data model.

When we talk about data warehouses, we have to have in mind that there is an open community that focuses on free data warehouse technologies with performances that are even better than the technologies in commercial use [3]. So which technology should be used and when? Namely, NoSQL database systems have certain advantages in terms of preprocessing or exploration of raw unstructured data, flexible programming languages running in parallel, analysis of provisional data etc. [14].

Companies such as Yahoo, Google and Facebook are collecting enormous amounts of data and relational (traditional) data storage models are just not sufficient in such scenarios. In order to avoid issues, these companies started to develop their own technologies to cope with large amounts of data, for example Hadoop platform ([8], [22]).

Hadoop has two types of data storage: Hive and HBase. „The Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL.“. [23] „Apache HBase is an open-source, distributed, versioned, non-relational database modeled after Google's Bigtable: A Distributed Storage System for Structured Data by Chang et al [8]. Just as Bigtable leverages the distributed data storage provided by the Google File System, Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.“[24].

As we said earlier, HBase is a column-oriented NoSQL representative. Data is stored in tables that have rows and columns, but columns are grouped in column families (one column family has more columns). It is important to notice that one cell can have more values over time and versioning is supported. One can look at the following picture (Fig. 3) to see column families, columns, rows and values:

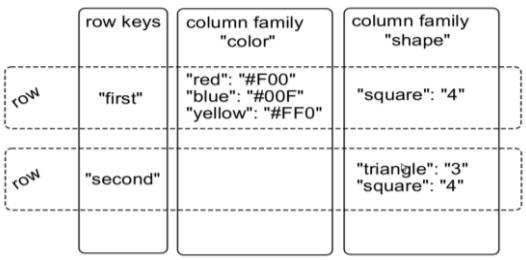


Figure 3. HBase table [9]

Since column values are stored together, it is easy to add new columns. For the same key, there are more rows with different values over time. In order to uniquely identify the row, we need to specify the version as well. HBase has some other advantages, as well as some drawbacks. Data does not have to be joined as we are used to do in SQL because data are denormalized. However, we do not have a system catalogue that one could use to find out interesting data about tables and other objects (we have to take care of it by ourselves).

3 Previous research

Converting data warehouse star schema to Hadoop Hive tables using Sqoop in order to test advantages and disadvantages was described in [7], and is also presented in Figure 2. below. Sqoop represents special developed tools that can migrate data to Hadoop. It „allows easy import and export of data from structured data stores: relation database, enterprise data warehouse and NoSQL Data store [7] “.

SAS is business analytics software. SAS data collection was used as a primary data source, from which the authors extracted data as a flat file without duplicates and stored data in star schema model. Both types are stored in the Hive data management system by the earlier mentioned Sqoop tool. Authors made a query performance and data compression test, which resulted with a difference in the query performance and data compress ratio.

Our idea was to use a similar approach, but instead of Hive we used the HBase data management system. Furthermore, we tried to implement a star schema in HBase table model.

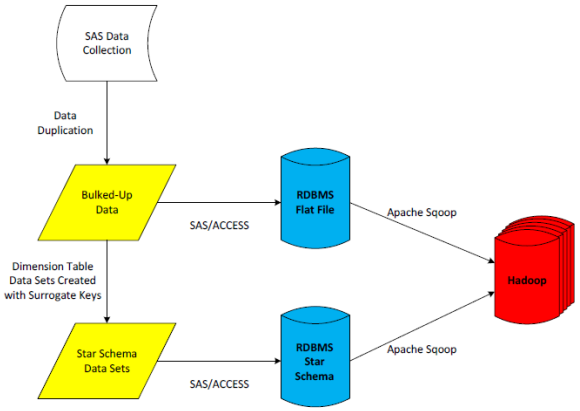


Figure 2. The Data load process [12]

The type of database which an organization require depends on its' needs [5]. Here we reached the discussion of advantages and limitations of column- and row-oriented databases. Row-oriented database is more efficient than column-oriented one when we have to select many columns of a single row, or when

a row can be selected with a single disk seek. Also, row-oriented databases are efficient when we have to insert a row within all column data at the same time [5], [11], [17], [6]. Otherwise, column-oriented databases are more efficient with the aggregation of many rows, but with a smaller subset of column data. Here we have a big plus on the analytics performance regarding the same type of value stored at one place, but under a limited number of situations CPU performance is not as good [5], [11], [17], [6]. Authors in [1] succeed to simulate a column store in a row store system via vertical partitioning and index-only plans, but simulation didn't yield a good performance because of high tuple reconstruction costs.

4 Implementation

4.1 Relational model of SMS messaging

In this section we want to present a small data warehouse scenario that is used for SMS message analysis. Nowadays, there are many services, such as parking, voting, content download, that are using SMS messages. Third world countries use „old" mobile technologies (SMS messages),, so there is still a need to build such data warehouses in order to serve the management with fresh and accurate data. Management wants to find out how many messages each service has per hour, which operator is the most represented, what is the amount of billed messages, what are the top five services by billed messages etc.

The star schema is given below. We have one fact and six dimension tables (Fig 4.). This data warehouse was implemented in MS SQL Server. Thus, each SMS message is stored in the „central table"– factTransaction. A fact table is connected with dimension tables and contains a foreign key towards each of the dimensions. In the Country table we can find data about countries (name, currency, Vat…); the User table contains data about the user of the service (mobile number). Data about the mobile operator is included in the Operator table (network code, name, …) and Service table includes details like Name, portfolio name etc. Finally DateH dimension gives us information about the date and time a message was received and submitted.

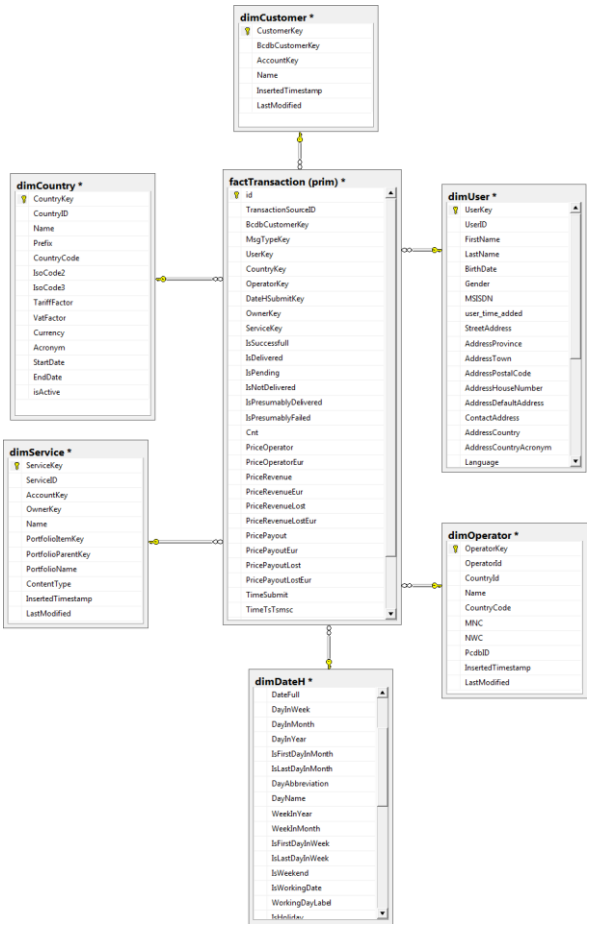


Figure 4. Data warehouse model

4.2 HBase implementation of SMS messaging

There are a few different ways to store data and make aggregations in HBase [10]:

1. Making HBase tables as external Hive tables - data can be accessed via HiveQL language here, but that is an unwanted additional step;

2. Writing MapReduce jobs and working with HBase data in the HDFS;

3. Using applications such as Crux, Phoenix, etc. in order to create HBase tables together with metadata needed for query processing.

In this paper, we applied the third approach, because we used HBase as a standalone installation, without a Hadoop file system. Moreover, we wanted to make analysis on HBase tables (not on Hive), so copying tables from one format to another was not applicable. Results that we got have been compared with SQL Server 2012 standalone installation.

Crux is a depreciated application. Since Apache took Phoenix in its project/application pool, we used Phoenix as the main application for querying HBase tables (Fig. 5). As we noted in the text above, HBase stored data without metadata, so there were some problems with aggregation and grouping. Phoenix

successfully avoids such problems and adds certain metadata.

We wanted to test both technologies with big data sets. When we inserted approx. 10000 rows in HBase, there was already a big difference in the performance. Finally, we made a test with one million data rows.

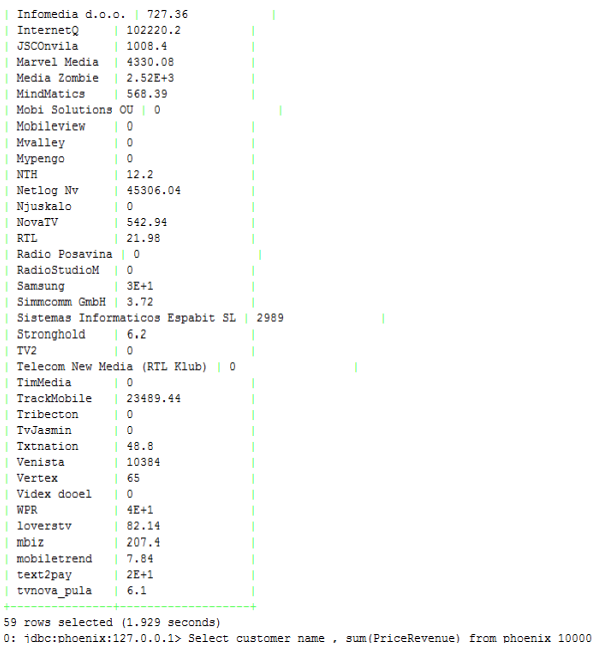
The data warehouse model was described above. We extracted and imported data to HBase tables by the means of a Phoenix Python script (psql.py) and we compared the performances for both solutions. Furthermore, we observed several scenarios of data model imported in HBase:

1. Denormalization of the entire data warehouse model in one flat table – facts could be joined with dimensions and imported to a single HBase table.

2. Second possible approach is to import dimensions and then the fact table, in that specific order. Dimension tables can have only one column family and this one would look different for the fact table.

3. The third approach would be to import each entity (fact, dimension) in a separate HBase table and then join the tables on the HBase querying level.

The second and the third option seem to be pretty complex for column-oriented databases. For example, joining tables in HBase and using relationships (foreign keys) for querying purposes, presents a problem with HBase querying framework. So we choose option one – flat data warehouse file.



```
| Infomedia d.o.o. | 727.36          |
| InternetQ        | 102220.2        |
| JSCOnvila        | 1008.4          |
| Marvel Media     | 4330.08         |
| Media Zombie     | 2.52E+3         |
| MindMatics       | 568.39          |
| Mobi Solutions OU | 0              |
| Mobileview       | 0               |
| Mvalley          | 0               |
| Mypengo          | 0               |
| NTH              | 12.2            |
| Netlog Nv        | 45306.04        |
| Njuskalo         | 0               |
| NovaTV           | 542.94          |
| RTL              | 21.98           |
| Radio Posavina   | 0               |
| RadioStudioM     | 0               |
| Samsung          | 3E+1            |
| Simmcomm GmbH    | 3.72            |
| Sistemas Informaticos Espabit SL | 2989 |
| Stronghold       | 6.2             |
| TV2              | 0               |
| Telecom New Media (RTL Klub) | 0       |
| TimMedia         | 0               |
| TrackMobile      | 23489.44        |
| Tribecton        | 0               |
| TvJasmin         | 0               |
| Txtnation        | 48.8            |
| Venista          | 10384           |
| Vertex           | 65              |
| Videx dooel      | 0               |
| WPR              | 4E+1            |
| loverstv         | 82.14           |
| mbiz             | 207.4           |
| mobiletrend      | 7.84            |
| text2pay         | 2E+1            |
| tvnova_pula      | 6.1             |
+------------------+-----------------+
59 rows selected (1.929 seconds)
0: jdbc:phoenix:127.0.0.1> Select customer_name , sum(PriceRevenue) from phoenix_10000
```

Figure 5. Phoenix query

Experimental results are given in Table 1. We created three different queries (some of them use GROUP BY clause) and we measured the execution time on both systems. One can see that MS SQL Server has much better performances than HBase:

Table 1. HBase vs. MS SQL Server

HBase Queries	SQL Server Queries	HBase exec. Time	SQL Server exec. time
Select id, TransactionSourceID, IsDelivered, TransportCost from phoenix_tbl	Select id, TransactionSourceID, IsDelivered, TransportCost from dbo.fact	11000ms, 1678341ms	116ms, 107168 ms
Select customer_name , sum(PriceRevenue) from phoenix_tbl group by customer_name	Select name , sum(PriceRevenue) from dbo.fact f inner join dbo.dimCustomer dc on f.CustomerKey = dc.CustomerKey group by name	1400ms, 24994ms	8ms, 243ms
select customer_name, sum(PriceRevenue) from phoenix_ tbl where isdelivered = 1 and customer_name like 'Im%' group by customer_name	Select name , sum(PriceRevenue) from dbo.fact f inner join dbo.dimCustomer dc on f.CustomerKey= dc.CustomerKey where isdelivered = 1 and name like 'Im%' group by name	60.6ms, 1397ms	2ms 36ms

5 Conclusion

In this paper we explored how NoSQL column-oriented representative called HBase can be used to implement the data warehouse. Although HBase is a column-oriented system, we expected that performances would be better, but that was not the case. SQL Server 2012 confirmed to be a robust solution with very good performances. Although similar attempts are made by other authors, they didn't use HBase system for data warehouse purposes; instead, they use Hive.

In the implementation phase, many problems occurred. Some tools that we tried didn't work and some were even not available (any more). So we can say that technologies that are not mature do show some peculiarities and that things are not always straightforward. We also discussed how star schema can be modelled in HBase and the best approach turned out to be the flat data model, i.e, denormalization of the entire data warehouse.

In the future, we would like to implement all tree approaches mentioned in the text above, in order to compare them. Moreover, using the column families in HBase could be an interesting subject for study as well.

6 References

[1] Abadi, D. J; Madden, S; Hachem, N. Column-Stores vs. Row-Stores: How Different Are They Really?, *ACM*, 2008.

[2] Adamson, C. *Mastering Data Warehouse Aggregates, Solutions for Star Schema Performance*, USA: Wiley Publishing, 2006.

[3] Awadallah, A; Graham, D. Hadoop and the Data Warehouse: When to Use Which, *Cloudera*, 2011.

[4] Ballard, C; Herreman, D; Schau, D; Schau, R; Kim, E; Valencic, A. *Data Modeling Techniques for Data Warehousing, IBM redbook*, IBM Corporation, International Technical Support Organization, 1998.

[5] Bhagat, V; Gopal, A. Comparative Study of Row and Column Oriented Database, *Fifth International Conference on Emerging Trends in Engineering and Technology*, 2012.

[6] Bhatia, A; Patil, S. Column Oriented DBMS an Approach, *International Journal of Computer Science & Communication Networks*, 2011.

[7] Bradley, C; Hollinshead, R; Kraus, S; Lefler, J; Taheri, R. Data Modeling Considerations in Hadoop and Hive, *SAS Institute Inc.*, 2013.

[8] Chang, F; Dean, J; Ghemawat, S; Hsieh, W. C. D; Wallach, A; Burrows, M; Chandra, T; Fikes, A; Gruber, R. E. Bigtable A Distributed Storage System for Structured Data, *Google Research Publication*, 2006.

[9] Chaudhuri, S; Dayal, U. An Overview of Data Warehousing and OLAP Technology, *ACM Sigmod Record*, 1997.

[10] Gruzman, D. Group by In HBase, http://stackoverflow.com/questions/9126489/, downloaded: April 21th 2014.

[11] Harizopoulos, S; Liang, V; Abadi, D. J; Madden, S. Performance Tradeoffs in Read-Optimized Databases, *VLDB*, 2006.

[12] Haque, A; Perkins, L. Distributed RDF Triple Store Using HBase and Hive, *University of Texas at Austin*, 2012.

[13] Inmon, H. W. *Building the Data Warehouse – Third Edition*, New York: John Wiley & Sons, Inc., 2002.

[14] Intel Corporation. Extract, Transform, and Load Big Data with Apache Hadoop, *Intel Corporation*, 2013.

[15] JSON, Introducing JSON, http://www.json.org/, downloaded: April 20th 2014.

[16] Kimball, R; Caserta, J. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*, Indianapolis: Wiley Publishing, Inc., 2004.

[17] Matei, G. Column-Oriented Databases, an Alternative for Analytical Environment, *Database Systems Journal vol. I*, 2010.

[18] Medjahed, B; Ouzzani, M; Elmagarmid, A. K. *Generalization of ACID Properties, Encyclopedia of Database Systems*, 2009.

[19] Ponniah, P. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*, New York: John Wiley & Sons, Inc., 2001.

[20] Prabhakar, A. Apache Sqoop: A Data Transfer Tool from Hadoop, *Cloudera Inc.*, 2011.

[21] Redmond, E; Wilson, J. R. *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*, Pragmatic Programmers, USA, 2012.

[22] Russom, P. Integrating hadoop business intelligence datawarehousing, *TDWI*, 2013.

[23] The Apache Software Foundation, Apache Hive TM, http://hive.apache.org/, downloaded: April 25th 2014.

[24] The Apache Software Foundation, Welcome to Apache HBase™, http://hbase.apache.org/, downloaded: April 25th 2014.